Evaluation of Star Identification Techniques

Curtis Padgett
Jet Propulsion Laboratory
California Institute of Technology

Kenneth Kreutz-Delgado
Department of Electrical and Computer Engineering
University of California, San Diego

Suraphol Udomkesmalee
Jet Propulsion Laboratory
California Institute of Technology

November 12, 1996

Abstract

1 Introduction

Accurate attitude information is a necessity for deep space missions. Without reliable orientation estimates, the ability of a spacecraft to send collected data back to earth, receive instructions, perform maneuvers and autonomously acquire targets is severely compromised. The large amount of time and money invested in space exploration projects requires reliable attitude determination systems in order to limit this possibility. The systems should have the ability to estimate the spacecraft's orientation with high precision and autonomously recover the attitude if it is lossed or found to be unreliable.

Fortunately there are several instruments that can supply attitude information to spacecraft control systems. Gyroscopes, sun, and star sensors are frequently integrated with onboard systems to provide feedback regarding spacecraft angular displacement. The signal estimation without prior information or slew'itlp, the sensor. Since star fields are typically visible in all orientations (for deep space missions) fixed sensors can image a section of sky, and determine the correspondence between the imaged stars and a catalog of reference stars stored onboard If no information is known about the current orientation, the entire onboard catalog can be consulted in order to determine the appropriate *matching*.

Algorithms that autonomously identify a star field with no a priori information regarding orientation are ideally suited for use in attitude initialization, support of star (rackers, *and* as a *backup* to primary attitude estimation systems. Current CCD sensors provide a relatively inexpensive way to image the sky and extract information about stellar locations and apparent brightness. A number of algorithms for star identification exist that can determine the correspondence between the viewed star field and a set of catalog stars in a known reference frame. Kosik[8] classifies a number of different techniques and provides a combinatorial argument with regard to their effectiveness along with some simulation results. However an extensive examination of how noise affects the performance and competence of star identification algorithms has not been carried out in a systematic way.

In this paper, we are interested in providing a useful framework for understanding and evaluating star identification techniques. It would be impossible (and unproductive) to do a comprehensive analysis of each existing star algorithms in the literature over the entire range of possible sensor configurations. Hence we limited our investigation to algorithms that met the following criteria:

- Fully autonomous (110 a priori attitude knowledge required).
- Representative of a general technique used in star identification.
- Return solutions promptly using non-specialized hardware and reasonable amounts of memory.
- General technique suitable for a small *fov*

We believe that *in* the future, spacecraft design will be toward smaller, less expensive systems with fewer sensors. Moving a fully autonomous star tracking and identification capability to either science or navigation sensors *eliminates* the *need* for both star trackers and sun sensors that are currently employed on most missions. The trends in star identification and tracking for space exploration will be toward fully autonomous, inexp*ensive*, fixed sensor, narrow field of view (*fov*) systems and our selection criteria are biased to reflect this. Many of the star identification algorithms in the literature are not fully autonomous and rely 011 other information to work properly. Some constrain the orientation of the sensor *fov* with the use of a sun sensor as in [?], or they use previous attitude estimates [1 2, 7, 17]. We are interested in techniques that perform adequately in a typical workstation environment and provide accurate solutions even when the size of the onboard database is very large as would be the case with small *fov*'s (< 8(1(f/7.(es).

In the next section the star identification problem is developed in greater detail and the strategies 11 s(,(1 to solve it, are classified into two general /1])] roaches. The algorithms we selected for more in depth evaluation are also described there. The simulation results of each of the selected algorithms are presented that reflect different levels and types of sensor noise. This measures the robustness of the algorithm's performance and provides insight into how well it might perform on more difficult problems. We conclude with a discussion that concentrates on the difficulties encountered with star identification for very large catalogs (i. e. small *fov*).

2 Problem Description

Put simply, the object of the star identification algorithm is to establish a correspondence between at least two points in the sensor reference frame and star locations from an onboard catalog in a standard reference frame. In general, this problem is quite difficult, however for most "natural", 2D euclidean problem instances there are a number of techniques that produce good results. In this section we divide the existing star identification algorithms into a small number of classes based on their formulation of the problem and the data structures used to solve it. In addition, we discuss the procedure for generating an onboard catalog from a star catalog and conclude with a more detailed description of the algorithms we have selected for evaluation.

2.1 Types of strategies for star identification

The most common approach in problem formulation for star identification is to treat the "known" stars as vertices in an undirected graph $\mathbf{G}$, with the angular separation between each pair of stars serving as the edge weights. The set of points extracted from the sensor also form an undirected graph $\mathbf{G}_s$. The problem of identification is then cast as finding a subgraph of $\mathbf{G}$ that is isomorphic to $\mathbf{G}_s$. Star identification algorithms that make use of this formulation typically carry a database of pre-computed distances between pairs of stars, or in some instances polygons (mainly triangles). The goal is then to construct an isomorphic subgraph(s) from the database with a similarly constructed sensor graph. The algorithms that form polygons or match groups perform star identification in this fashion [8, 20, 19, 1, 7, 5, 6].

The other major formulation of the star identification problem is to associate a pattern with each star in the onboard catalog. The pattern found for an individual star should only make use of its neighboring sky characteristics. This will allow similar patterns to be constructed from a the sensor image. The goal of identification is to find the catalog pattern that is closest to the sensor pattern generated using sensor star s_j. More formally, given a location vector $\vec{v_{c_i}}$ for a star c_i in the onboard catalog, the goal is to find some function f that generates pattern p_i. The input to f is a neighborhood region around the location vector that consists of the expected sky for that region as viewed by the sensor (for clarity, we will assume that $f(\vec{v_i})$ is sufficient to indicate the region about star c_i). Ideally f should have the property that:

$$match(c_i, s_j) \leftrightarrow \| f(\vec{v_{c_i}}) - f(\vec{v_{s_j}}) \| \leq \epsilon < \| f(\vec{v_{c_k}}) - f(\vec{v_{s_j}}) \|, \forall k \neq i$$

The ϵ parameter defines a neighborhood around each catalog pattern in order to prevent matching a pattern generated by a sensor star with no corresponding catalog star.

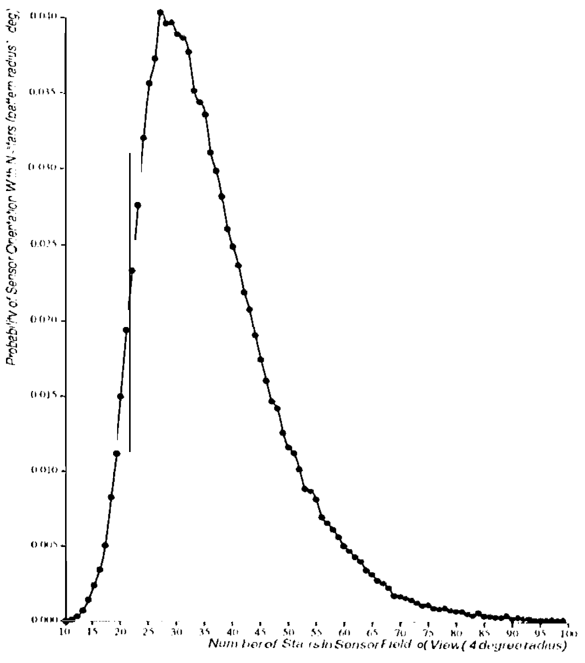


Figure 1: Probability distribution for viewing N stars in an orientation for a 4 degree radius *fov* for stars down to magnitude 7.5.

In essence, this approach attempts to derive a signature, unique for each star in the catalog, that can be found when viewing the section of sky with the star in it. Matching seeks to find the closest catalog star signature to a given sensor star's signature. A database for this approach consists of the patterns for each onboard catalog star, typically stored in a lookup or hash table for efficiency. Algorithms employing this approach include [13, 10, 12].

2.2 Onboard star catalog

The onboard star catalog C, is simply a subset of the stars whose locations are known in a standard reference frame. Each entry i in C will typically contain a location vector $\vec{v}_i$ and an apparent brightness value b_i. Deciding which particular stars to include in C; is a decidedly difficult task. The selection of the stars is dependent on the range and *fov* area of the sensor, the algorithm used for identification, and the degree of confidence required in identifying an arbitrary star field. The goal is to have a roughly uniform distribution of stars across the sky contained in C so that the chosen identification algorithm performs competently over all possible spacecraft attitudes.

To generate a suitable C, we need to find the set of stars that are likely to be imaged by the sensor. Using a standard star catalog, we can find those stars whose corrected apparent brightness is greater than the minimum sensitivity of the sensor. The stellar magnitude value given in typical star catalogs[11, 9] is not necessarily the same value measured during imaging; it usually requires some normalization with respect to the sensor. Leibe [10] demonstrates how to estimate this value for a given set of sensor parameters. Not all of the stars brighter than the sensor's minimum sensitivity should be included in C. Stars which are close to each other when imaged on the focal plane (e.g. binary stars) tend to interfere with each other during localization and brightness determination. Other stars have variable brightness. The location and brightness measurements for these two classes of stars are unreliable and are removed. The remaining stars that are usually detected by the sensor, serve as the set from which C is generated.

The distribution of stars is not uniform over the sky. Some orientations will have many more stars that the sensor is likely to image than others (see Figure 1). Typically, the star identification algorithm will require some constant number of stars $k > 2$ at every orientation to identify a star field unambiguously. The value for k depends on the algorithm and the expected level of noise in the image. Little or no noise allows for tighter threshold tolerances resulting in fewer distance/brightness matches between star pairs or star patterns. In these cases, k can be reduced without reducing the probability of identifying the star field.

If we wish at least k stars in every orientation, then some criteria will have to be used to select the best stars for C. Typically the brighter stars in any given sensor *fov* are more reliably imaged and extracted than are dimmer stars. A simple scan over the entire sky can be conducted with the remaining stars. The brightest stars at each orientation in the sensor *fov* can be placed in C. A similar strategy can be employed with the objects imaged by the sensor in order to locate likely elements of C in cases where the image contains many stars.

3 Selected algorithm descriptions

We chose to evaluate three of the existing star identification algorithms using the extraction scheme for C as described above. Two of the algorithms selected treat identification as a subgraph isomorphism problem while the third looks to establish a correspondence based on a best matching pattern. While this is not a complete survey of the existing techniques, the algorithms selected are representative. Many of the remaining algorithms in the literature can be considered a variation of one of the techniques we selected. Most of the others are either unsuitable for fully autonomous star identification or would be prohibitively expensive with respect to computing resources, especially with large star catalogs. The algorithms selected include a triangle matching approach, one based on forming match groups or pole stars, and finally a pattern based approach using an oriented grid.

3.0.1 Triangle algorithm

The triangle identification algorithm is a variant of perhaps the most common currently used technique. This approach is similar to that outlined by [6] and attempts to match uniquely a triangle configuration on the sensor with an isomorphic triangle from the onboard catalog. One or two methods are typically used in identifying isomorphic triangles: either SAS (side-angle-side) or SSS (side-side-side). In our implementation, we chose to use SSS which requires only a single threshold, ϵ_d, to establish edge matches. The other technique, SAS, is also appropriate (and perhaps more

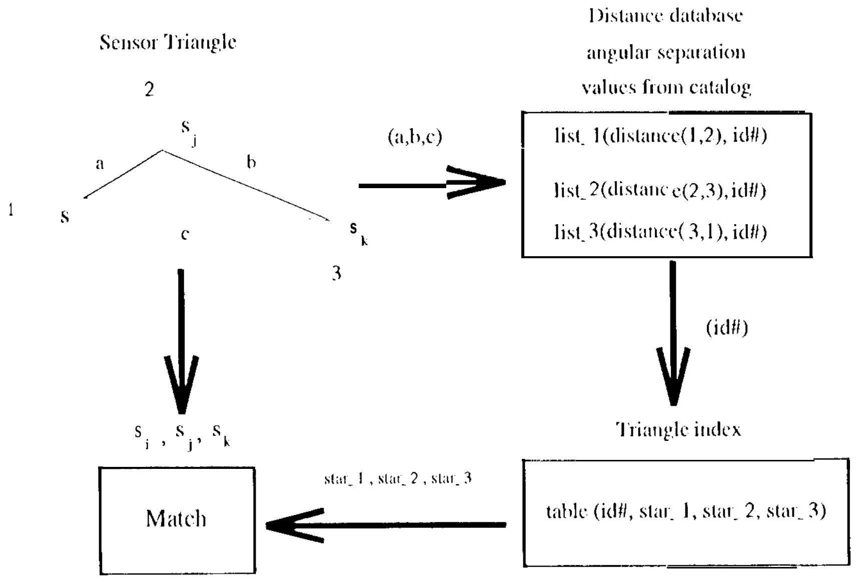


Figure 2: The triangle star identification algorithm. See text for description.

flexible) but would require two different threshold parameters.

To achieve efficiency in triangle identification, those triangles in C that are viewable in a single sensor orientation must be identified initially and stored in a database. Otherwise the cost of generating all possible triangles in C would be incurred for each identification. For small *fov*'s and large Cs, this would be prohibitive. In our algorithm we implement this database as three arrays sorted by angular distances. Each array holds the distance of a single leg of a triangle and a unique number that identifies the triangle.

To generate the database, a consistent labeling of each triangle in C is required. Our labeling is based on the angular separation values of an arbitrary catalog triangle, $\triangle c_i c_j c_k$ The angular separation values between each pair of vertices are calculated and ordered, $a < b < c$. The vertices can then be uniquely labeled such that vertex 1 shares both a and c, vertex 2 shares a and b, and vertex 3 shares b and c. For each triangle in C, a unique identification number n is given and entered along with $a, b,$ or c into one of three lists (list_1, list_2, or list_3 respectively). After all triangles from C are entered, the lists are sorted by distance so that ranges can be extracted in $O(\log N)$ time using a binary search[15]. This distance database is useful for extracting the catalog triangle identification numbers based on leg length.

A table called the triangle index is used to hold the vertices associated with a particular catalog triangle. The triangle identification number serves as the table location. At each location, the catalog star for each vertex is identified. Thus the distance database serves to identify the particular triangle and the triangle index locates the corresponding catalog stars. Identification is based on determining which catalog triangle best corresponds to a given sensor triangle. Figure 2 provides an overview of the significant steps in the triangle identification algorithm.

A triangle from the sensor image, $\triangle s_i s_j s_k$ is labeled in the same manner as the catalog triangles. Each leg size is used to index the appropriate section of the distance database and find all triangle identification numbers within $\pm\epsilon_d$. A counter associated with each catalog triangle is incremented for each identification number of a matched leg. A sensor triangle that matches three edges with a catalog triangle is considered isomorphic. The individuals stars of the catalog are then paired with the labeled sensor stars using the triangle index.

There are two major difficulties with the triangle algorithm that still need to be addressed. The first problem deals with how the sensor triangles are formed. From Figure 1 it is obvious that if triangles are made for each set of three stars in the sensor, an inordinately large number of sensor triangles will go through the matching process. However, the number of stars in $\mathbf{C}$ is about k per orientation by construction. Hence we can limit the number of stars from which sensor triangles are constructed to the brightest αk. The α term helps insure that most of the sensor stars that are also in $\mathbf{C}$ are available to be matched. Otherwise measurement noise in brightness determination might inadvertently exclude stars that we want to include.

The second problem has to do with evaluating the matched sensor/catalog star pairs. Ideally we would like to correctly identify as many sensor stars as possible since this would typically provide a better estimated attitude [16]. This suggests that all the matched sensor triangles be used in generating the correspondence (instead of the best matched triangle, for instance). The problem is that some of the sensor triangles matches may be spurious (i.e. they are paired with the wrong catalog triangle). This occurs when a sensor triangle is matched with more than one catalog triangle or a sensor triangle not in $\mathbf{C}$ is inappropriately identified with a catalog triangle. These cases arise because noise and measurement error require looser tolerances forcing a larger value for ϵ_d while large numbers of catalog triangles require tighter tolerances to insure uniqueness. In any event, some method must be used to reduce the possibility of incorrectly identifying an imaged section of sky with a spurious triangle match.

Our solution is to initially remove any matched catalog triangle if the apparent brightness of any star at a vertex is not within $\pm\epsilon_b$ of the brightness at the corresponding sensor vertex. While this does entail the use of another parameter (dependent on sensor accuracy), it should serve to make each of the catalog triangle more unique resulting in fewer spurious matches. Our final evaluation step is to look at the locations of the stars identified. If there are no spurious matches, all the identified stars should be within a single *fov* diameter. We can use this idea to search the identified locations for the area with the greatest number of stars. Provided that the sensor *fov* is small with respect to the entire sky and spurious matches occur uniformly over it, more than three identified stars in a single area is good evidence that the matches are not spurious. Those identified stars not in the area are removed as spurious. The identification portion of the triangle algorithm is summarized below.

1. Find the brightest αk objects from the sensor image

2. Generate a list of $C\binom{\alpha k}{3}$ sensor triangles for identification.

3. For each triangle, label the vertices, and increment a triangle identification counter for each catalog edge that is within $\pm e_d$ of the appropriate sensor edge.

4. For each counter with value three, determine if the brightness for each labeled sensor vertex is within $\pm e_b$ of corresponding catalog star. If they are, put the star pairs in a matched list.

5. If there are no items in the matched list indicate failure. Otherwise check to see that all catalog stars are within the same sensor *fov* diameter. If they are not and a largest grouping of catalog stars exists within the same *fov*, consider this group the identified stars. If no largest group exists (they are all of equal size), indicate failure.

3.0.2 Match Group algorithm

The match group algorithm is another common approach to star identification that seeks to find an isomorphic subgraph in the catalog for a specific configuration of sensor stars. This technique was first discussed by Kosik [8] and further refined by others [19, 1]. Our implementation and description of the match group algorithm is essentially as described by van Bezooijen [20] though we did not fully utilize all of the verification steps reported there. Instead, we simply check for the largest set of stars in each matched group that meet the distance tolerance. While other verification steps could certainly be employed that may marginally improve the identification rate, we are mainly interested in the performance and competence of the underlying algorithm so it is our intention to have the verification steps be as similar as possible. The verification step employed by the match group algorithm is more powerful than those used by the triangle and grid algorithms, but the identification strategy requires additional checking to preserve its integrity.

The idea in this algorithm is to find a catalog star (*pole star*) whose distances to its neighbors is most similar to a given sensor star and its neighboring stars (*satellites*). The configuration to be matched is shown in the upper left panel of Figure 3. The distance from the pole star to each of the other sensor stars is used to index a table of distances between pairs of stars in C whose distance is less than the *fov* diameter.

The entries in the table consist of a distance value and pointers to the two catalog stars. The table entries within $\pm e_d$ of the sensor distance are located using the strategy described for the triangle algorithm (binary interval search). The brightness values of the catalog stars are used to determine the appropriate match between the sensor segment and each matched catalog segment. If the brightness values are within $\pm e_b$ for both endpoints, the segment is considered matched. It is possible that the catalog segment could match in two ways, in which case both matches need to be included.

Once all of the matched catalog segments are extracted from the table, the algorithm determines the largest match gy'oil]). Since the pole star in the sensor appears in each of the segments used in indexing the table, its corresponding catalog star should appear in the set of matched segments for each satellite star. Due to measurement error, the addition/loss of stars from the star field, and noise, this may not be the case but large match groups do provide good evidence that a correspondence

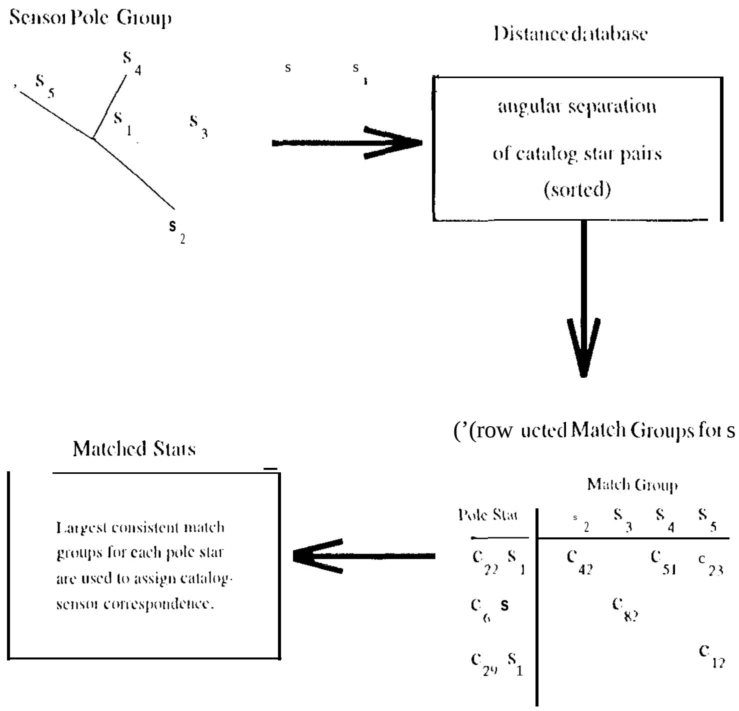

Sensor Pole Group
s_4
s_5
s_1
s_3
s_2
s
s_1
Distance database
angular separation of catalog star pairs (sorted)
Constructed Match Groups for s
Match Group
Pole Star
s_2 s_3 s_4 s_5
c_{22} s_1 c_{42} c_{51} c_{23}
c_6 s c_{82}
c_{29} s_1 c_{12}
Matched Stars
Largest consistent match groups for each pole star are used to assign catalog-sensor correspondence.

Figure **3:** The match group star identification algorithm. See text for description.

may exist. The lower left panel in Figure 3 demonstrates a method for keeping track of the match groups.

Each match group is identified by the catalog star c_i, paired with the sensor pole star s_p. For the endpoint associated with c_i, the catalog star identification number is entered into match group i under its corresponding sensor star. After all the catalog segments have been entered, the match groups for the particular choice of pole stars is complete. As was the case for the triangle algorithm, this procedure is then repeated with other sensor stars so that ok stars are processed.

After finding the match groups for each of the sensor pole stars, the next step is to determine which of the large match groups are consistent. This can be done by verifying that the satellite stars in the match group and the associated catalog stars are within tolerance. Of course many of the match groups will be redundant as they are the same section of sky centered on different pole stars. As these are easily identified, they are removed and simply treated as a single match group. The largest consistent match group is then considered the appropriate correspondence. The algorithm is summarized below.

1. Find the brightest αk objects from the sensor image.

2. Make each of the αk objects a pole star, calculate the distance between it and its neighbors, and find the segments in the distance database which are the same size $\pm\epsilon_d$.

3. For each sensor group, if possible, label the vertices of each catalog segment so that the

brightness values of the sensor and catalog stars are within $\pm 1\, \epsilon_b$ of each other. Register each segment in the appropriate match group.

4. Find the largest consistent match group by verifying the distance relationships amongst the satellite stars in the group.

5. If there is no largest match group or he size of the largest match group is not sufficient the algorithm returns failure. Otherwise the match group provides the correct correspondence between the sensor and catalog.

3.0.3 Grid algorithm

The last algorithm that we have selected forms a pattern around a given star and searches a set of patterns generated from the stars in **C** in order to determine the best match. The algorithm described here can be viewed as an extension of the algorithm described y Liebe [10]. In that algorithm, a pattern is generated from a given star and its two nearest neighbors. The pattern consists of the angular separation between the given star and each neighbor and the angle between them. Each measurement is normalized and concatenated together to form a single number which is then used to index a hash table. The entry in the table gives the appropriate correspondence. In the grid algorithm [13], the pattern is formed from the entire surrounding star field thus encapsulating more information. Figure 4 demonstrates pattern formation in the grid algorithm.

The patterns are generated by initially picking a star s_i, finding the closest neighbor s_n, outside some radius r, and orienting a grid on the coordinate system defined with s_i at the origin and s_n indicating the positive x-axis. All stars within the pattern radius r_p, are than projected onto the grid. This process is shown in Figure 4 panels A-C. The grid is of size g x g and is typically at a much lower resolution than is the sensor. The resultant pattern is simply a bit vector with a 1 in each grid cel that has a star and 0 in cells that don't have a star (see panel D Figure 4). This pattern is the signature for star s_i

To determine the best match with the catalog patterns, a bit comparison is made with each pattern in the catalog. The agreement between two patterns is found by finding the total number of on grid cells that are shared. Between sensor pattern p_s and catalog pattern p_c, this value is expressed as:

$$\sum_{i=1}^{g*g} p_{s\,i} \wedge p_{c\,i}$$

where $\wedge$ is logical and of the two bits.

The closest matching catalog pattern p_c, is considered to have the largest sum from the above expression. If the value of the sum is above a threshold m, sensor star s is paired with catalog star i. The value of m is set to make it highly unlikely that two random patterns with the average number of stars a would have a sum greater than m. The probability of getting more than m matches must be quite small (i.e. m must be relatively large) since the sensor pattern is compared with thousands or perhaps even tens of thousands of catalog patterns.

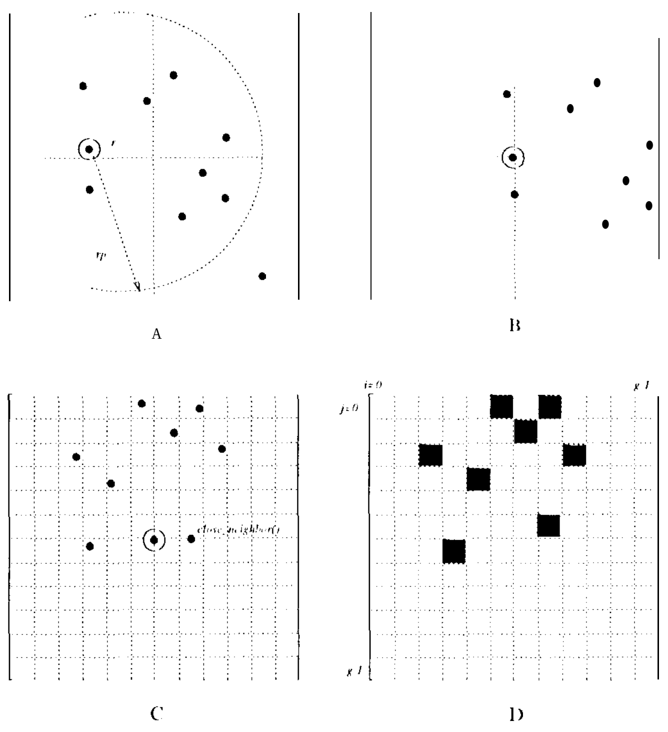


Figure 4: Pattern formation in the grid algorithm. See text for description.

Implementation of a bit vector and the comparison of the sensor pattern with each catalog pattern is more inefficient than is necessary. The bit-vector is typically quite sparse, enough so that simply keeping track of each grid cell with a star in a list uses less memory. This alternate representation also suggests a convenient storage mechanism for the catalog patterns. Instead of keeping an array of catalog patterns, each pattern can be entered into a table indexed by grid cell location. A table entry is simply a pointer to a catalog star that has the corresponding grid cell in its pattern. The identification process for a sensor pattern is to read the entries in the table at each grid location with a star, increment a counter associated with each individual star in the table entries, and keep the catalog star index of the counter with the greatest value. The star pattern associated with this star is then the closest matching catalog pattern. If the number of matches is greater than a threshold parameter (described earlier as ϵ) a match occurs between the sensor star and catalog star at the center of each respective pattern.

As was the case in the first two algorithms, we are only interested in forming patterns for sensor stars that have a corresponding catalog star so that we only select the αk brightest stars in the sensor field for identification. Unlike the first two algorithms however, there is no excessive computational demand in using all the stars in the sensor field to form the patterns. Utilization of all stars makes each of the patterns more unique, providing greater assurance that the matched star pairings are correct. This however, now requires that the catalog patterns include the entire surrounding star field that the sensor is expected to see (not just other catalog stars). The following is a brief outline of the identification portion of the algorithm (see [13] for additional details):

1. Find the brightest αk objects from the sensor image.

2. For each object, find the nearest neighboring star, orient the grid and form a pattern using the entire sensor field.

3. Find the closest matching catalog pattern. If the number of matches is greater than m, pair the sensor star with the catalog star associated with the pattern.

4. Perform a consistency check similar to the triangle algorithm; look for the largest grouping of identified stars within the *fov* diameter of each other. If the size of this group is greater than 1, return the pairings in this group. If no largest group exists, or no identifications occur, report failure.

4 Simulation environment

Star identification takes place in a typical machine vision context An initial raw image is received from the sensor and processed to extract a set of objects and their properties. Unlike most problems in machine vision however, the properties of stars typically used for identification (location and apparent brightness) are well defined and difficulties arising from occlusion, scaling, and object appearance are not of major concern. Stars (due to their great distance) can be treated as point sources of light from a location on the unit sphere. Using an idealized model of a sensor (see Figure 5), the location (x, y), of an imaged star on the focal plane can be determined from the star's unit vector $\vec{v}$ in the sensor's reference frame as follows:

$$x = f * v_1/v_3$$

$$y = f * v_2/v_3$$

The coordinate (x, y) is the location point of the object on the sensor plane and f is the effective focal length in the same units. The star locations are derived by translating the spherical coordinates found in standard stellar catalogs to a rectangular system and rotating the sky to the appropriate sensor orientation.

The minimum sensitivity of the sensor is used to decide if a star in the *fov* is bright enough to be imaged. A high range term (low stellar magnitude) indicates the extent of useful brightness measurements on a star. A star brighter than this value is imaged but the reported value is simply equal to the maximum brightness. Also included in the sensor configuration are noise terms specifying location and brightness accuracy, expected deviation in effective focal length and its corresponding mean value.

To provide realistic sensor locations however, the accumulated error from imaging and object extraction should be reflected in the input coordinates. The amount of error that occurs in the individual locations arises from a number of different sources. In the lens/sensor system dispersion,

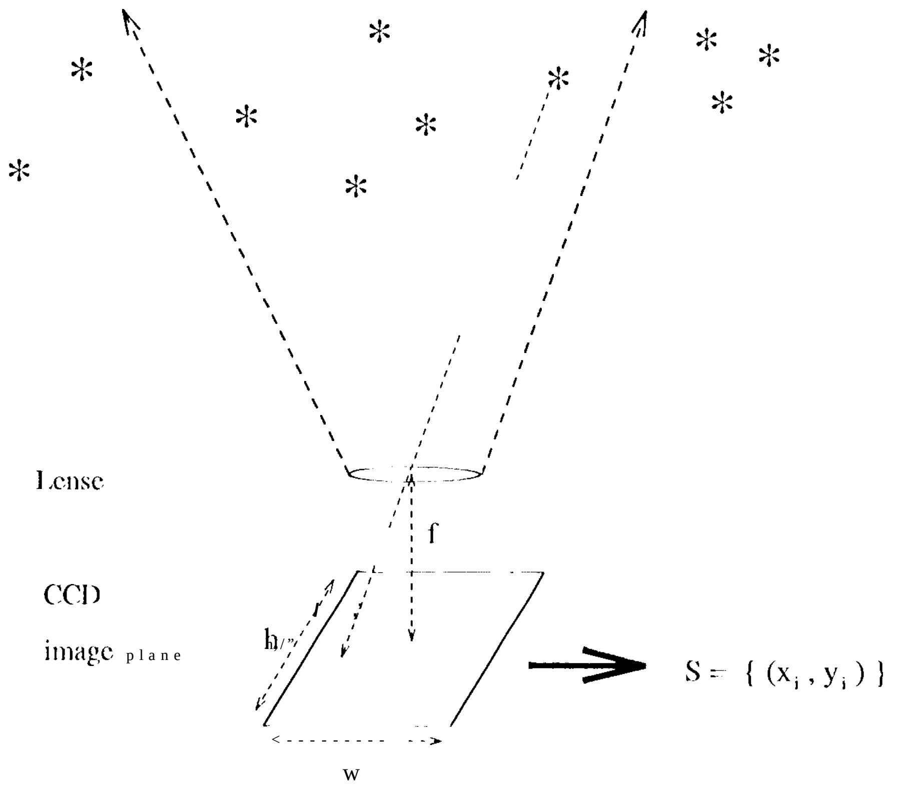


Figure 5 : Idealized sensor geometry for locating stars on the sensor focal plane. Value f is the effective focal length.

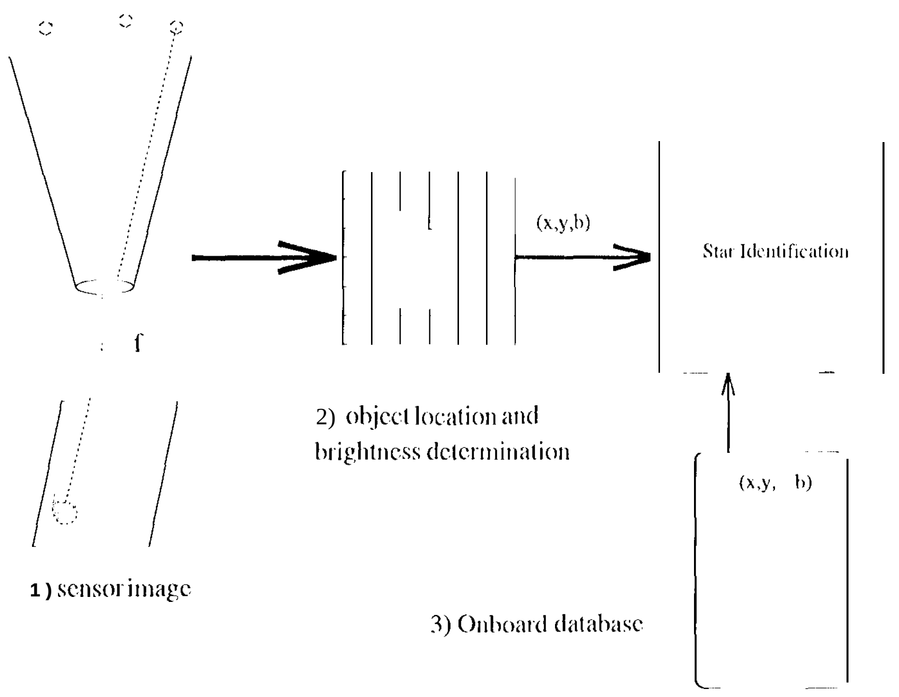


Figure 6: Image capture and preprocessing involved in providing input to the star identification algorithm. The numbered steps indicate areas where noise is introduced into star identification

aberration, distortion, focal length deviation, etc. tend to perturb how the light falls on the sensor plane that if uncorrected would change the point to point correspondence between the image plane and the true world state. Most of this type error can be corrected during calibration and would be sensor specific [9], However the change in focal length due to thermal expansion and contraction is a likely source of systematic error from this system that is common to all star sensors. We can expect deep space probes with many years of continuous operation to experience this type of noise with some certainty. The sensor locations of the stars are easily derived with the use of the above equations given the change in focal length.

A change in focal length will also affect the overall brightness measurement for imaged stars. The point location on the sensor plane is usually taken as the centroid of a blob of activation over a number of pixels (see Figure 6). The blob is generated by slightly defocusing the lens (hence the effective focal length of the idealized sensor) so that photons from a light source will strike a number of neighboring pixels in the sensor. The activation of each pixel in the blob can be used to construct a curve and the apparent brightness estimated from that. A change in focal length will affect the size of the blob, introducing systemic error in brightness determination. This type of error is easily incorporated in the simulation environment by adding or subtracting a fixed value to each of the imaged star's brightness.

In addition to the systematic error generated by the lens/sensor system, there is a considerable amount of noise introduced by this system and during the preprocessing of the sensor image. Diffuse

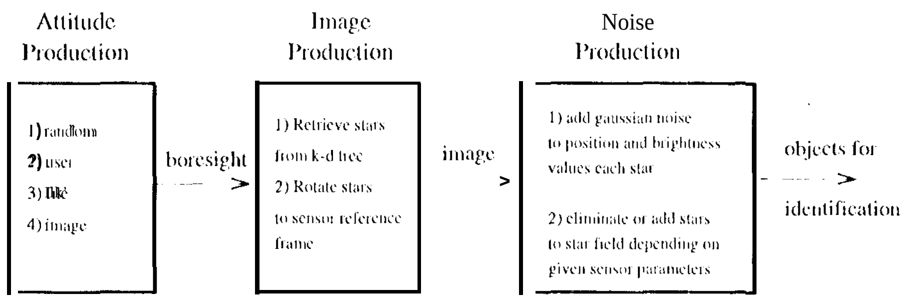

Attitude Production
1) random
2) user
3) file
4) image
boresight
Image Production
1) Retrieve stars from k-d tree
2) Rotate stars to sensor reference frame
image
Noise Production
1) add gaussian noise to position and brightness values each star
2) eliminate or add stars to star field depending on given sensor parameters
objects for identification

Figure 7: Generating a set of stars for identification Systematic error is introduced in the box labeled Image Production while Noise Production acids simple gaussian error to lint], the brightness and location valu es.

lighting, radiation, planetoids, and limited sensor precision cause the addition or deletion of expected viewable objects, and undermine the ability of the extraction p rocess to accurately n ıea sure the brightness and location of a star. The star ext raction process itself also int roduces additional error with the use of thresholds in local izing the blob and in calculating the brightness curve. Imaged stars near the threshold face the possibility of being eliminated as sensor objects if none of the pixels in their blobs are above the threshold. This results in the loss of stars that are expected to be viewed. The opposite situation is also possible in that stars slightly below the minimum sensitivity of the sensor could be imaged providing additi onal, unexpected objects [3, 14]. The calculated location of the star on the sensor plane as determined by the previous equations is used as the expected location and is perturbed using an error parai neter (in pixel units of sta ndard deviation) and Gaussian noise. A si milar process occurs for the measured brightness values. Steps 1 and 2 of Fig ure 6 show where noise is introduced during the imaging process.

The other source of noise (shown in the figure as step 3) is due to the onboard catalog itself. The values incorporated in a star catalog typically have a small amount of error [1 1] due to stellar motion, measure n ient processes, visual magnitude to sensor brightness, unidentified stars, etc. For large *fovs*, the error introduced by t he catalog is relat ively small when compared to other sources of noise, however for larger catalogs and smaller narrower sensors, this noise may be significant and is included in the software environment. Catalog parameters indicating the deviation for the location and bri ghtness of individual st ars are use '(1 to generate eac i of the onboard cata logs u sec I by the star identification routines. T1 ıe stellar cat alog used to generate the sky images is considered the true sky as in a Monte Carlo method.

Figure 7 shows an overview of the sky simulation process. Star identification algorithms are tested u1 ıder a great variety of changing noise conditions so that some measure of an algorithm's robustness can be learned. In addition, a set of parameters for a single system co1 ıfiguration ca1 ı be tested against a number of different algorithms or their parameter setti ıgs to discover which particular algorithm or setpoints are most effect ive. Our evaluation of the selected star identification algorithm makes use of this methodo logy in the results reported in the next section.

5 Results

In this section we describe the performance and competence of each selected star identification algorithm. Performance issues are concerned with the amount of memory the catalog and data structures consume as well as the time it takes to determine the correspondence. For our purposes, algorithm competence deals with the overall robustness of the algorithm in handling noise and examining what situations lead to failure.

The "true" sky for the simulation environment is from a catalog provided by the Autonomous Feature and Star Tracking project (AFAST) at the Jet Propulsion Laboratory. The catalog contains nearly 36,000 stars down to a stellar magnitude of 8.0. The sensor configuration simulated in these experiments was designed to make maximal use of the available stars. The simulated CCD used a 512x512 pixel array (11.8 mm) with each dimension spanning a full 8 deg arc. The minimum sensitivity of the sensor was set at 7.5 units stellar magnitude and the reported values ranged to a maximum of 2.5 units. The star field images for such a system contained approximately 45 stars on average. The *fov* area was about 64 square degrees while the effective focal length was 84.2 mm.

The onboard catalogs C for the star identification algorithms were constructed as described previously using the values of the "true" sky perturbed by a small amount of gaussian noise for both the location (1 arc second standard deviation) and the apparent brightness (0.1 units stellar magnitude standard deviation) values. Two different size catalogs were generated, one requiring at least 5 stars per *fov* and the other 9 stars. The resultant catalogs contained 7548 and 11,901 stars, or about 9 and 14 stars on average per image, respectively.

5.1 Performance

For core (onboard) catalogs of both sizes, we generated a database suitable for each of the selected algorithms. Figure 8 provides the total number of data structures required to build the supporting database for each algorithm. The total memory for each algorithm is somewhat dependent on implementation details, but assuming that a star reference can be made with 2 bytes while a float requires 4, a fairly accurate estimation can be performed.

The triangle algorithm requires three floats for each leg of a triangle in the database, plus a reference to the triangle (4 bytes) and each of its stars for a total of 22 bytes per triangle. The total memory consumed by the algorithm also includes the cost of storing each star in C, which requires two floats to specify each location and a float to indicate apparent brightness. The actual memory used in identification is quite small when compared to the database, so that we need not consider it. The memory for the smaller core catalog totals nearly 3Mb while the larger catalog uses about 12Mb.

The match group algorithm uses much less memory for database structures as compared to the triangle algorithm. It has traded the static memory allocation strategy of the triangle algorithm's database for a much larger scratch space in active memory in order to form the groups. The size of the scratch space needs to be as large as the worst case condition (or employ some complicated

Algorithm	|C| = 7548	|C| = 11,901
Triangle	134,000 triangles	555,000 triangles
Match Group	66,000 pairs	166,000 pairs
Grid	7548 patterns	11,901 patterns

Figure 8: Size of supporting datastructures for each algorithm

checking) which depends on the number of matches per edge and the total number of edges involved in the matching. These values vary with the parameter settings of the algorithm as determined by the noise level and the size of the catalog. If the algorithm is operating in a noisy environment, the average number of matches can be as high as 4500 per edge. Forming pairs with the 10 brightest stars from the sensor image results in using almost the entire catalog as a pole star in a match group so that the scratch space ends up being as large or larger than the space for C. The entire memory then consists of 2 star references per edge and its distance, plus two times the size of C (we are assuming that the scratch space and the space required by the core catalog are roughly equal). For the smaller core catalog this works out to be about 0.7Mb, while the larger catalog consumes over 1.6Mb. The non-linear increase in space occurs here due to the change in core star density from 9 stars per *fov* for the smaller catalog to 14 for the larger one. This results in the total number of edges per *fov* increasing by about 2.5.

The memory consumed by each pattern in the grid algorithm depends on the average number of stars included in the pattern. For the sensor system and parameters we are using, this works out to be about 25 stars. As each column in the lookup table represents a pattern grid cell, only a single reference to the central star of the pattern is required. As was the case with the triangle algorithm, the active memory is quite small as compared to the catalog and lookup table, so we do not include it The tots] memory for the smaller core catalog is nearly 0.5Mb while the large catalog requires a little over 0.7Mb.

The other major element in performance is the amount of time required to return the correspondence or indicate failure. The time per identification is dependent on a number of factors: the size of C, the amount of noise, the individual algorithm parameter settings, and how many stars from the sensor image are tested for membership in C. The subgraph strategies (match group and triangle algorithms) are extremely sensitive to the particular thresholds used, and times vary considerably depending on their settings. To make the comparison between the algorithms, we report the time for conditions that are likely to be experienced.

The acceptable tolerance for distance matching for either triangle edges or star pairs was set at the smallest value (i.e. the value that generated the fewest distance pair matches on average) that achieved the highest identification rate when the location of a star on the sensor was perturbed with gaussian noise of 1 pixel standard deviation around the true location. This resulted in about 1250 matches per edge with the smaller sized C, and over 3000 matches on average for the larger catalog. The actual noise environment used in the test was maintained with a location error of 0.5 pixels

Algorithm	|C| = 7548	|C| = 11,901
Triangle	1.8	8.3
MatchGroup	1.6	7.4
Grid	0.04	0.12

Figure 9: Average time in seconds to identify a star field. The location error is 0.5 pixels and brightness error is 0.3 units stellar magnitude (plus systematic error incorporated in the catalog).

and brightness deviation of 0.3 units stellar magnitude. For the triangle algorithm, matching was conducted using only the 7 brightest stars on the sensor for a total of 56 triangles. The match group algorithm made use of the top 10 edges for a total of 45 pairs while the grid algorithm composed patterns with the 15 brightest stars as the central star for the smaller catalog, and 27 brightest for the larger sized C. The grid algorithm made use of a 40x40 grid.

Figure 9 presents the results of the time experiments. The table clearly shows the cost inherent in expanding the catalog. Both of the subgraph strategies appear to have a quadratic growth rate hidden in the matching strategy so that very large catalogs could be problematic (the number of sensor stars evaluated in both algorithms was held constant). For the grid algorithm, the increased number of stars evaluated is likely to explain most of the growth. Indeed if we evaluate 27 sensor stars in the smaller sized C the runtime per identification increases from 0.04 seconds to 0.09 seconds. A strictly proportional increase in runtime for the larger catalog would work out to about 0.14 seconds so that the reported time of 0.12 seconds reflects the time hidden by the lookup table.

5.2 Competence

The second focus of our results is on how well the algorithms perform under various noise conditions. We concentrate on exploring how the identification rate changes as different types of noise levels increase. Our methodology is similar to that found in [4]. The algorithm parameters are fixed and a single noise source is increased while the others are held at typical levels. The change in identification rate demonstrates the algorithm's underlying robustness.

Our first two tests look at each algorithm's best identification rate which returns the matching in less than two seconds. This time limit is certainly near the outer envelope for responding to crucial mission operations and represents a reasonable amount of time for this size problem. The distance parameters for the triangle and match group algorithms were set as described in the previous section (about 1.5 pixels). The brightness threshold was set at 1.0 units stellar magnitude. Both the triangle and grid algorithms made use of the smaller catalog in order to meet the required time constraint. The triangle algorithm generated triangle using the 7 brightest stars (56 triangles) on the sensor while the match group algorithm exploited the brightest 10 stars (45 edges). Due to its superior time performance, the grid algorithm used the larger core catalog, and attempted to match the top 27 brightest stars on the sensor. The grid size was fixed at 40x40 during all the tests.

Figure 10 and 11 show the identification rate for the three algorithms as the instrument accuracy

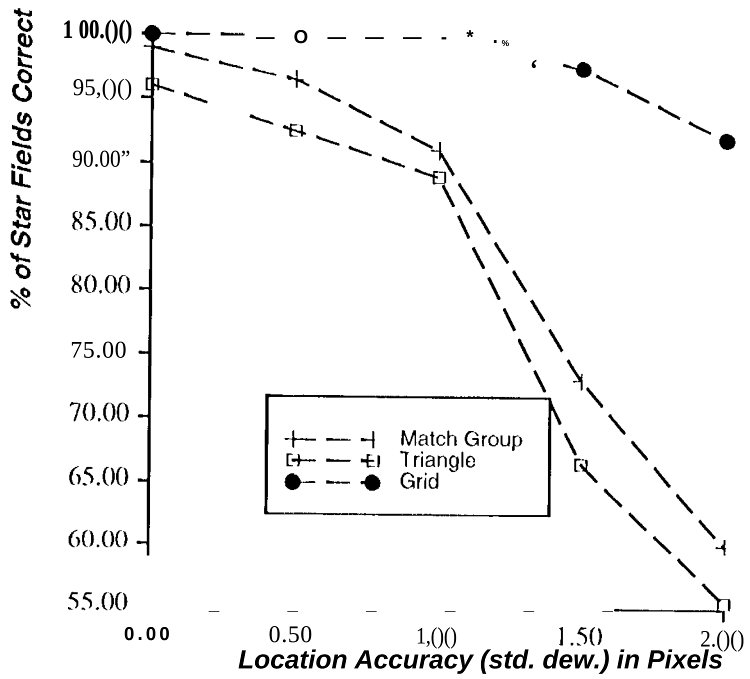


Figure 1 (): The three curves represent the percentage of star fields identified over 500 random orientations of the sensor for each plotted point. The brightness error was held constant with a standard deviation of 0.3 units stellar magnitude. The standard deviation for the location error is shown along the horizontal axis in pixels.

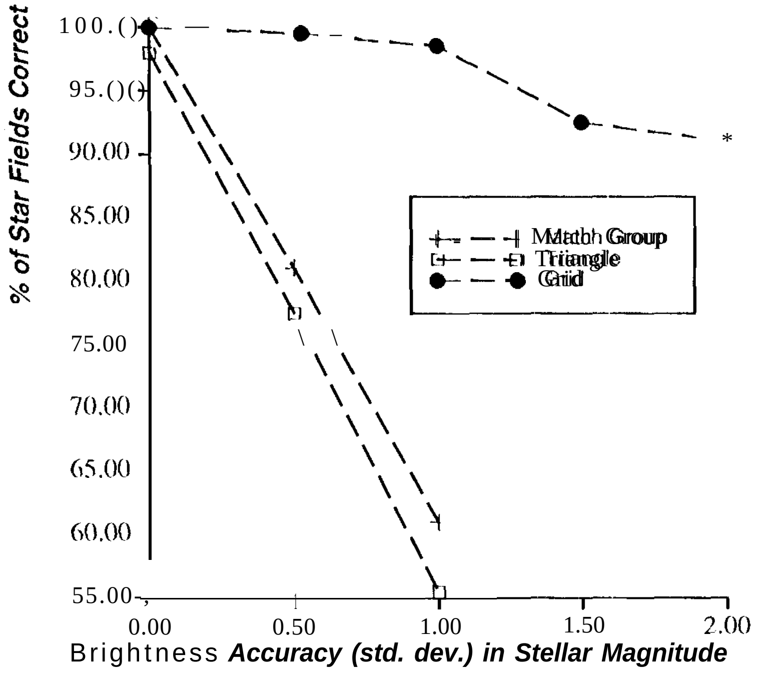


Figure 11: The three curves represent the percentage of star fields identified over 500" random orientations of the sensor for each plotted point. The location error was held constant with a standard deviation of 0.5 pixels. The standard deviation for the brightness error is shown along the horizontal axis in units stellar magnitude.

decreases. Each algorithm was presented with a sequence of 500" star fields selected randomly to identify at each shown noise level. The brightness values reported to the algorithm from the sensor were perturbed with gaussian noise so that some stare that would typically be visible were lost (i.e. below sensor minimum sensitivity). About 2.6 stars per image disappeared, on average for 0.3 units standard deviation in stellar magnitude with the average increasing linearly up to 13.0 stare per image for 2.0 units. In addition to the stars lost, some stars that would not normally be seen were imaged so that 1.8 extra stars appeared per image on average at 0.3 units deviation and 3.8 stars at 2.0 units. For Figure 10 the brightness deviation was maintained at 0.3 units stellar magnitude while Figure 11 shows the identification rates with the location deviation held at 0.5 pixels.

The false positive rate (indicating a wrong correspondence) for the two subgraph matching algorithms varied linearly with the position accuracy. At 0.5 pixels deviation the percentage of false identifications was 1.5% and increased to nearly 15% for 2.0 pixels. For high noise levels in brightness accuracy, nearly 1/5 of the non-identified orientations were mis-identified. Of the mis-identified cases however, close to 1/8 identified the correct section of sky but failed to pair the sensor/catalog stars correctly. Over the range of tests discussed in this section, the grid algorithm experienced only two false positives.

The relative steep drop in the identification rate (as compared to the grid algorithm) for the

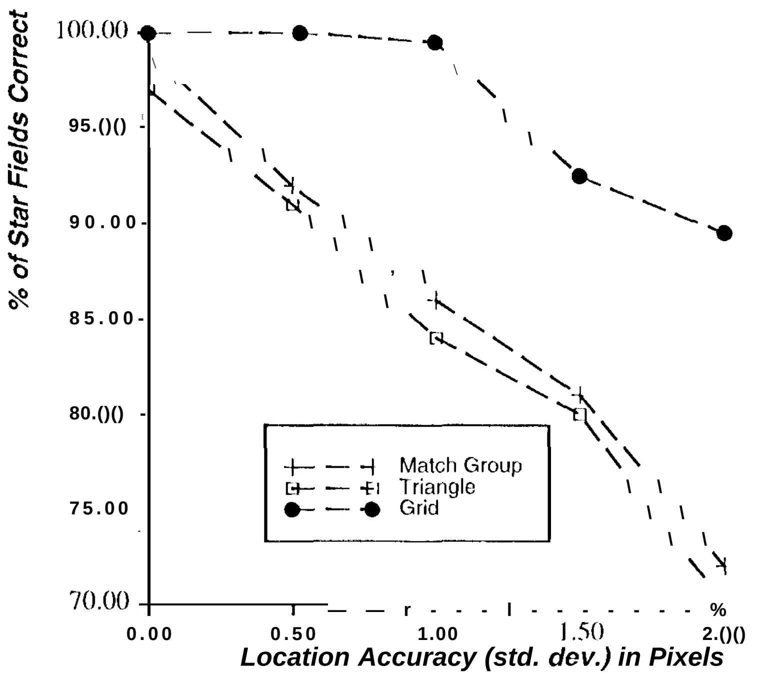


Figure 12: The three curves represent the percentage of star fields identified over 500" random orientations of the sensor for each plotted point. The brightness error was held constant with a standard deviation of 0.3 units stellar magnitude. The standard deviation for the location error is shown along the horizontal axis in pixels. The triangle and grid algorithms use an adaptive threshold and the grid algorithm uses the smaller catalog while checking only the brightest 1 () sensor stars.

triangle and match group algorithms stems from (in part) the selection of the distance threshold. If we use an adaptive threshold for these algorithms, there is a significant improvement in the curves. The stability of the grid algorithm's identification rate is dependent to a degree on the number of sensor star patterns formed and the size of the core catalog. If we use the smaller catalog and limit the number of sensor stars checked to be comparable with the other algorithms (1 ()), a noticeable decrease in the algorithm's robustness is apparent Figure 12 shows how the respective identification rates change as the accuracy of the location information varies.

Our final result examines how systematic error affects the identification rate for each of the algorithms. To accomplish this we assume that the effective focal length of the sensor system has changed as could happen if the system is subjected to substantial thermal stress. As in the previous experiments, other sources of noise are held at nominal values, in this case 0.5 pixels deviation for location and 0.3 units stellar magnitude for brightness. The effective focal length is changed from the calibrated value (84.2 mm) in 1mm increments. Figure 13 shows the identification rate for each algorithm (above 50%) as the effective focal length changes by about 3.6%.

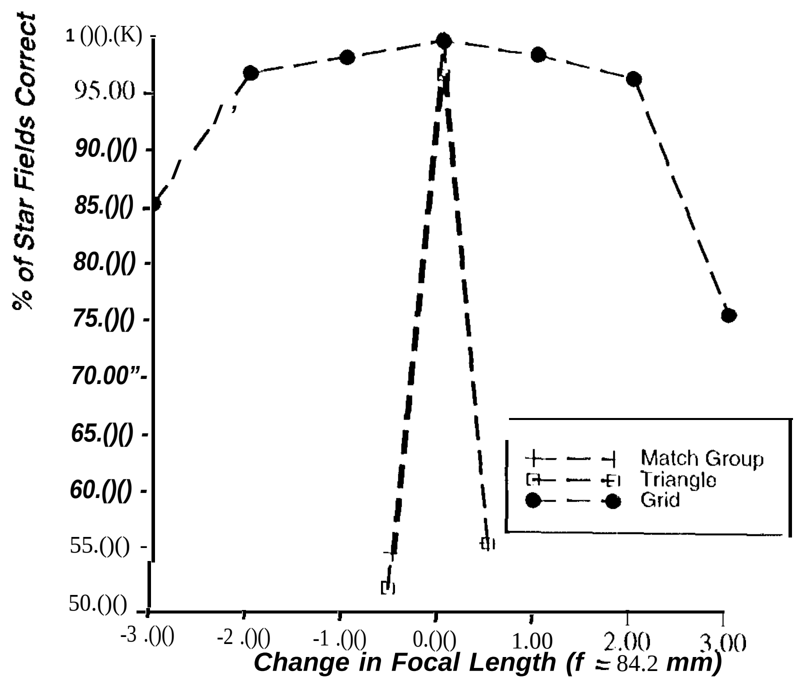
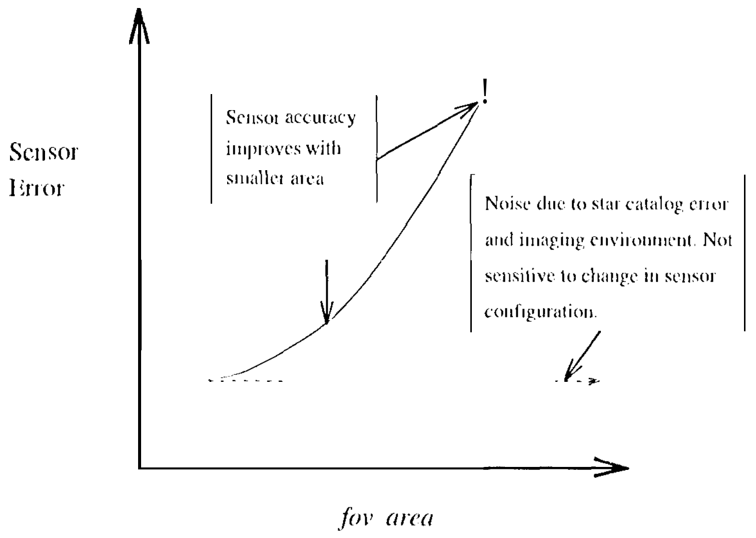


Figure 13: The three curves represent the percentage of star fields identified over 500 random orientations of the sensor for each plotted point. The location and brightness error were held constant with a standard deviation of 0.5 pixels and 0.3 units stellar magnitude. The horizontal axis plots 1mm changes in the effective focal length of the sensing system (calibrated at 84.2 mm).



Sensor Error
Sensor accuracy improves with smaller area
Noise due to star catalog error and imaging environment. Not sensitive to change in sensor configuration.
fov area

Figure 14: The sensor error does not decrease proportionally with the decrease in *fov* area. For a given sensor, a fixed amount of noise limits the absolute accuracy of the measurements requiring additional evidence to be accumulated for a sensor image to maintain the same star identification rate.

6 Discussion

The results from the previous section showed that the star identification strategies based on distance matching between pairs of stars are considerably less robust than the pattern matching algorithm tested. The change in identification rates over the different noise conditions provides us with a direct measure of how unique each star field is within the underlying pattern space that the particular star identification algorithm is working in. A *robust* problem formulation provides maximum separation between the individual subgraphs or patterns given an arbitrary onboard catalog. This aids in achieving a graceful degradation of star identification competence for high noise environments and allows for a higher density onboard catalog required for smaller *fov*'s.

For the more accurate, smaller *fov* star sensors envisaged for future space missions, denser (more stars in C per *fov*) onboard star catalogs are a necessity. For such systems, systematic error due to accuracy limitations of the stellar catalogs, optics, etc. will become a larger portion of the noise environment. This will result in proportionately greater noise levels for the more accurate system. The expected operating range in terms of pixel error of the star identification algorithm will increase, meaning that the area of interest in Figure 10 will be to the right of the current range (about 0.5 1.0 pixels). In order to achieve similar identification rates for a single sensor image, more information will need to be extracted from the star field (see [18] for an alternative approach using a sequence of sensor views). Figure 14 provides a conceptual depiction of the problem.

It is important then to ask how well different star identification strategies will work in a higher noise environment. It is not our contention that the results of the previous section reflect the highest identification rate for the respective algorithms. Certainly a large number of additional changes could be made to each algorithm, and to C in order to increase its particular identification rate for a given noise level. Using the more extensive checking described by [19, 4], for instance, results in about a 2% improvement in the identification rate for the match group algorithm with noise levels below 1 pixel deviation and a 1% improvement for higher noise levels (adaptive threshold). However it is our experience that many of the verification techniques used to weed out spurious matches in the star identification algorithms are independent of the particular strategy used to make the correspondence, and result in only a marginal improvement in the identification rate. They are essentially a grab bag of tricks to aid in determining which of the identified star matchings are most likely and can easily be adapted to any of the techniques.

Our purpose in using a similar consistency check in all the algorithms rather than a more comprehensive routine as in [19] is to understand how well each basic star identification strategy works, and to determine the underlying robustness of the algorithm. The function of the routines we are examining is to formulate the matching in such a way that the star patterns or subgraphs developed from the onboard catalog are as unique as possible, yet are observable in a high noise environment or a dense onboard catalog. This ensures that a correspondence between a given sensor image and the correct pattern or subgraph is possible and that other patterns or subgraphs will match with less probability and in fewer numbers.

The major distinction between the two star pair distance matching strategies and the grid algorithm involves the uniqueness of a match, not the type of verification technique used. A single cell location in a sensor pattern for the larger core catalog used in the simulation will generate about 250 matches (60% fewer using the smaller catalog) on average (catalog patterns that share that particular cell), of which at least 249 are wrong. The triangle and match group algorithms on the other hand, match more than 5 times as many edge pairs on average, over 1250 matches per edge using the smaller core catalog and a distance threshold of about 1.5 pixels.

The spurious matches (where spurious in this instance is defined as any match other than the correct one) can be thought of as match "noise", which effectively hides the true signal (match), thus making identification more difficult. Additional evidence is required to unmask the true signal, and this is provided by forming the relevant structures used in identification either triangles, match groups, or patterns. However the amount of additional evidence gathered depends (mostly) on the size of the structure. The triangle algorithm's base size is limited to three. Match groups, in principle, provide a much larger structure with which to make decisions, but it is typically limited by performance considerations to less than 15 (10 in our simulation). In actual practice, the average size of the largest match group was much closer to the size of the triangle structure between 4 and 5. The size of the sensor structure for the grid algorithm is closer to 20 while the average number of matches for a correctly identified pattern is nearly 13.

From these observations it is quite easy to see why the identification rate of the grid algorithm degraded much Slower than did the other two algorithms. The grid algorithm performs identification in an environment that generates fewer spurious matches (lower match noise), and provides significantly more evidence in the form of a larger underlying structure by making use of stars not in C. The two subgraph matching strategies are less robust because of the large amount of match noise and the incorporation of less evidence during identification. The obvious solution is to identify more triangles for the triangle algorithm or to attempt to match larger groups for the match group algorithm.

However this solution is problematic at best. Performance considerations alone weigh heavily against such an approach. As Figure 9 shows, the use of the larger core catalog for these algorithms results in a fourfold increase in time to realize a solution, and a near threefold increase in the size of the database. Even with the larger catalog, however, the change in identification rate is quite small and appeared to drop even more dramatically than when using the smaller catalog.

Clearly the reason the performance degrades so dramatically for both of these algorithms is the nonlinear growth rate associated with the matching operation (the grid algorithm has a linear rate of growth). If n is the average number of core stars per *fov*, the triangle database grows at $O\left(n^3\right)$ while the size of memory required by the match group algorthm grows at $O\left(n^2\right)$. A near 50% increase in the size of C resulted in close to a 320% increase in match noise for the same noise environment (1.5 pixels). This greatly slowed the identification time making it much km costly to increase the number of sensor triangles or star pairs tested since this grows nonlinearly as well. If we let a be the number of sensor stars tested (i.e. the brightest a stars), the growth rate of edge pairs is $O\left(a^2\right)$ and is even worse for triangles. The only benefit then for increasing the size of C is to make it more likely that the actual sensor stars have a corresponding match in C, but this proved to be quite marginal.

7 Conclusion

We have reviewed two major strategies employed in star identification. Our results and analysis indicate that a pairwise matching technique employed in the most prevalent routines may have difficulties in the next generation of autonomous star sensing systems. The large, dense onboard catalogs required for such systems inevitably generate a considerable number of false matches making identification more difficult. Pattern matching techniques such as those used in a relatively simple grid algorithm [13] seem to hold more promise.

References

[1] M. Baldini, A. Foggi, Benelli G., and A. Mecocci. A new star-constellation matching algorithm for satellite atitude determination. *European Space Agency Journal*, 17:185 198, 1993.

[2] A. Batten. Iso ground attitude determination during operations. In *Proceedings of the ESA Symposium on Spacecraft Flight Dynamics*, pages 259 264, Darmstadt, Germany, 1991.

[3] R. Blue, D. Chang, and E. Dennison. Performance testing of oca/llnl star tracker. Technical Report EM 343-1304, Jet Propulsion Laboratory, 1993.

[4] L. DeAntonio, S. Udomkesmalee, J. Alexander, E. Blue, E. Dennison, G. Sevaston, and M. Scholl. Star-tracker based, Fill-Sli}', autonomous attitude determination. *SPIE Proceedings*, 1949:204 215, 1993.

[5] D. Gottlieb. Star identification techniques. In J. Wertz, editor, *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Co., MA, 1978.

[6] E. Groth. A pattern-matching algorithm for two-dimensional coordinate lists. *Astronomical Journal*, 91:1244 1248, 1986.

['i] J. Junkins, C. White, and D. Turner. Star pattern recognition for real time attitude determination. *Journal of the Astronautical Sciences*, 25:251 27(1, 1977.

[8] J. Kosik. Star pattern identification aboard an inertially stabailized spacecraft. *Journal of Guidance, Control and Dynamics*, 14(2):230 235, 1991.

[9] B. Lasker, C. Sturch, B. McLean, J. Russell, H. Jenker, and M. Shara. The guide star catalog: I. astronomical foundations and image processing. *Astronomical Journal*, 99(6):2019 2178, 1990.

[10] C. Liebe. Pattern recognition of star constellations for spacecraft applications. *IEEE Aeronautics and Electronic Systems Magazine*, June 1992.

[11] S. McLaughlin. Skymap star catalog version 3.5, 1989. National Sapce Science Data Center, Goddard Space Flight Center.

[12] F. Mill'tagll. A new approach to point-pattern matching. *Astronomical Society of Pacific*, 104:301 307 , April 1992.

[13] C. Padgett and K. Kreutz-Delgado. A grid algorithm for star identification. *submitted IEEE Aerospace and Electronics Systems*, July 1994.

[14] D. Purll, N. Gradmann, and M. Bollner. The rosat star tracker- flight experiece. In *Proceedings of the First ESA International Conference on Spacecaft Guidance, Navigation and Control Systems*, pages 551- 556, Noordwijk, Netherlands, 1991.

[15] R. Sedgewick. *Algorithms*. Addison-Wesley Pub. Co., Reading, MA, 1988.

[16] M. Shuster and S. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4:70 78, 1980.

[17] '1'. Strikwerda and J. Junkins. Star pattern recogn ition and spacecraft attitude determination. Technical Report ETL- 0260, Engi neer To pographic Laboratories, 1981.

[18] S. Udomkesmalee, J. Alexander, and A. Tolivar. Stochastic star identification. *Journal of Guidance, Control, and Dyna nics*, Ja n .-Feb. 1995.

[19] 1{. van 1 3ezooijen. *Automated Star Pattern Recognition.* PhD thesis, Stanford university, 1989.

[20] R. van Bezooijen. A star pattern recognition algorithm for autonomous attitude determination. In *Automatic Control in Aerospace; IFAC Symposium*, pages 51 58, Tsukuba, Japan, 1989.

[21] J. Wertz. *Spacecraft Attitude Determination and Control.* D. Reidel Publishing Co., MA, 1978.